



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

15

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/847,063	04/30/2001	Ming Zhou	GEI-001US 29083	4555
21718	7590	12/02/2005	EXAMINER	
LEE & HAYES PLLC SUITE 500 421 W RIVERSIDE SPOKANE, WA 99201			RUTTEN, JAMES D	
			ART UNIT	PAPER NUMBER
			2192	

DATE MAILED: 12/02/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/847,063	ZHOU ET AL.
	Examiner	Art Unit
	J. Derek Ruttent	2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 07 October 2005.
- 2a) This action is **FINAL**. 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-4,7-11,13-15,18-24,26,32-35,39-43,49,51,52 and 55 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 1-4,7-11,13-15,18-24,26,32-35,39-43,49,51,52 and 55 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date. _____.
3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date <u>10/7/05</u> .	5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)
	6) <input type="checkbox"/> Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 7 October 2005 has been entered.

2. Applicant's arguments, see page 10 of the Remarks, filed 7 October 2005, with respect to the rejection of claims 1-4,7-11,13-15,18-24,26,32-35,39-43,49,51,52 and 55 under 35 U.S.C. § 103(a) have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new grounds of rejection is made in view of U.S. Patent 6,802,059 to Lyapustina et al.

Response to Arguments

3. Applicant essentially argues on pages 5 and 6 (labeled as argument "A") that deficiencies exist in "Gawk 3.1 new feature list" by Robbins. In light of further arguments presented on page 8 regarding interpretation of the phrase "in a same operation", the Robbins reference is withdrawn from the rejection. Thus, the arguments on pages 5 and 6 are moot.

4. Applicant essentially argues on pages 6-10 (labeled as argument "B") that the art of record, particularly the reference by Sun, fails to "remove the locale-sensitive content as claimed". The first argument presented is that use of the gettext() function simply wraps around

the locale-sensitive content, but does not remove it (page 7 lines 18-20). This argument is bolstered by page 10 lines 2-4: “‘Removing’ should be given its plain meaning, indicating that the locale-sensitive content is literally *moved* from the document”. Broad interpretation of the nature of the removal of locale-sensitive content as explained in paragraph 7 of the 10/20/04 Office Action is maintained. However, in the interest of the advancement of the application, Applicant’s present arguments have been interpreted more narrowly to require that a removal of locale-independent content would necessitate the removal of the original sequence of characters present in Sun’s text string. As such, Applicant’s arguments are convincing. However, a new ground of rejection is made in view of U.S. Patent 6,802,059 to Lyapustina et al.

5. It is noted that Applicant argues on page 8 lines 19-22 that the phrase “in a same operation...may encompass any series of manual and/or automatic operations of any nature, potentially even including one or more intermediary operations not identified in claim 1.” This definition appears to broaden the scope of the claim beyond the plain meaning of the phrase. Since such an operation as so defined could encompass *any series* of manual and/or automatic operations of *any* nature, the Robbins reference is no longer required and is withdrawn. The Sun and Lyapustina references can now be interpreted as providing this limitation since it teaches extraction, removal, and substitution in a series of operations.

6. Applicant argues on pages 10-12 (labeled as argument “C”) that there is no motivation to combine the references since they are based on incompatible methodologies. As noted above, the Robbins reference has been withdrawn, and those arguments directed toward Robbins are moot. In this case, Hinks discloses a system including a resource file that provides locale-sensitive content. An editor is provided to translate from one locale to another. Sun teaches that

instead of being edited directly, locale-sensitive content can be “supplemented” with a function call that scans a separate file for content. The use of a separate file provides a level of abstraction that allows the locale-sensitive content to be edited without having to rebuild the entire package. While Hinks and Sun provide two different implementations, Sun’s teaching does not change the principle operation of Hinks’ Export/Import module, or translation table. Sun simply teaches the use of function calls to provide a layer of abstraction. Thus, Applicant’s arguments are not convincing.

Oath/Declaration

7. The oath or declaration is defective. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by application number and filing date is required. See MPEP §§ 602.01 and 602.02.

The oath or declaration is defective because:

While it acknowledges the duty to disclose as defined in 37 CFR 1.56(a), it does not state that the person making the oath or declaration acknowledges the duty to disclose to the Office all information known to the person to be material to patentability as defined in 37 CFR 1.56 as a whole.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

9. Claims 1-4, 13-15, 18-24, 26, 32-35, 41-43, 49, 51, 52, and 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record U.S. Patent 5,678,039 to Hinks et al. (hereinafter referred to as "Hinks"), in view of prior art of record "OpenWindows Developer's Guide: Xview Code Generator Programmer's Guide" by Sun Microsystems (hereinafter referred to as "Sun") further in view of U.S. Patent 6,802,059 to Lyapustina et al. (hereinafter "Lyapustina").

As per claim 1, Hinks discloses:

A computer-implemented method comprising:

analyzing a computer-servable document written for a particular locale See column 3 lines 7-9:

First, the sources are parsed by the Export/Import module to a translatable format.
extracting locale-sensitive content from the document while leaving locale-independent elements in the document See column 3 lines 1-4:

The Export/Import module itself includes a parsing engine to extract strings and translatable information from application programs.

Also column 3 lines 39-43:

In either instance, the underlying program code (i.e., the code which the programmer has written to carry out the functionality of the program) has remained untouched by the process.

storing the locale-sensitive content in a data structure separate from the document (column 3 lines 13-16).

Hinks does not expressly disclose the removal of locale-sensitive content or the substitution of a locale-sensitive content with a function call.

However, in an analogous environment, Sun teaches manipulation of *locale-sensitive content* See page 100:

If you then use the GXV code generator to generate the C source code, the interface's text strings get linked with the gettext() or dgettext() function call. These calls are used to look up the strings in other languages, or locales.

Locale-sensitive content is removed through the process of linking the function calls in the code. The function calls use the text strings as keys to a text database of translations for other locales. As they are used as keys to a database and are no longer directly available to the original source code, the strings are no longer locale-sensitive.

substituting, in a same operation as the extracting and removing, a function call in place of associated locale-sensitive content in the document See page 100 as cited above:

...the interface's text strings get linked with the gettext() or dgettext() function call. *the function call being configured such that, when executed, the function call obtains the locale-sensitive content from the data structure* (page 98 under the header “gettext() and dgettext() Routines”).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun's function call substitution in Hinks' localization system. One of ordinary skill would have been motivated to include nothing specific to one's language and culture in software development and to provide features that facilitate translation of text into other languages.

Hinks and Sun do not expressly teach *removing* locale-sensitive content. However, in an analogous environment, Lyapustina teaches that locale-sensitive content can be removed and substituted with a unique identifier string. See column 4 lines 18-22:

Upon identifying each string, the conversion mechanism generates a unique macro string as a substitute for the original string. The conversion mechanism then substitutes the unique macro string for the identified string in the source code of the computer program.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Lyapustina's unique macro string with Sun's function call. One of ordinary skill would have been motivated to avoid the use of hard coded strings (See Lyapustina column 1 line 64 – column 2 line 7).

As per claim 2, the above rejection of claim 1 is incorporated. Further, Hinks discloses *wherein the analyzing comprises examining each line of code in the document and based on the code, identifying the locale-sensitive content* (column 3 lines 1-4).

As per claim 3 the above rejection of claim 1 is incorporated. Further, Hinks discloses *wherein the locale-sensitive content comprises natural language text* (column 3 lines 28-31).

As per claim 4 the above rejection of claim 1 is incorporated. Further, Hinks discloses *wherein the locale-independent elements comprise source code and formatting data* (column 3 lines 31-34).

As per claim 13, Hinks discloses:

A method comprising:

automatically compiling a computer-servable document written for a particular locale to extract ... any locale-sensitive content (column 3 lines 1-4, and 7-9 as cited in

the above rejection of claim 1), *the compiling producing a compiled document with locale-independent elements* See FIG. 3 element 377 and column 8 lines 30-34:

Alternatively, a Resource Compiler 365, again such as Borland's Resource Workshop®, may be employed to re-compile the Translated Resource Files 360 and bind those compiled resources back into the target program, now shown as Translated Program 377.

storing the locale-sensitive content in a form that can be translated to other locales See column 3 lines 15-22:

From there, Export/Import (EXPIMP) module parses the resource file into a Translation Table, which is typically stored as a database table. The Translation Table encapsulates all the information that is known or can be derived from the various resources and stores them in a format which may be utilized by various editors.

Also column 3 lines 53-55:

In this fashion, changing a product from one locale to another can be reduced to the simple process of swapping out resource files.

Hinks does not expressly disclose the removal and substitution of a locale-sensitive content with a function call.

However, in an analogous environment, Sun teaches *substituting a function call in place of the locale-sensitive content in the compiled document* (page 99 under the header “xgettext and msgfmt Utilities”), *the function call being configured such that, when executed, the function call obtains and embeds the locale-sensitive content back into the compiled document* (page 98 under the header “gettext() and dgettext() Routines”).

All further limitations have been addressed in the above rejection of claim 1.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun's function call substitution in Hinks' localization system. One of ordinary skill would have been motivated to include nothing specific to one's

language and culture in software development and to provide features that facilitate translation of text into other languages.

As per claims 14 and 15, all limitations have been addressed in the above rejections of claims 3 and 4, respectively.

As per claim 18, the above rejection of claim 13 is incorporated. Further, Hinks discloses *retrieving, at runtime, the compiled document and populating the compiled document with the locale-sensitive content to reconstruct the computer-servable document that can be served to the particular locale* (column 3 lines 50-55).

As per claim 19, the above rejection of claim 13 is incorporated. Further, Hinks discloses *wherein the locale-sensitive content is translated in to a second version for use in a second locale* (column 3 lines 35-36). All further limitations have been addressed in the above rejection of claim 18.

As per claim 20, Hinks discloses:

compiling a computer-servable document written for a particular locale to extract ... any locale-sensitive content, the compiling producing a compiled document with locale-independent elements (column 3 lines 1-4, and 7-9 as cited in the above rejection of claim 1; also FIG. 3 element 377 and column 8 lines 30-34 as cited in the above rejection of claim 13); .

storing the locale-sensitive content (column 3 lines 15-22 as cited in the above rejection of claim 13); and

at runtime, retrieving the compiled document and populating the compiled document with the locale-sensitive content (column 3 lines 50-55 as cited in the above rejection of claim 18).

Further, Hinks discloses obtaining the associated locale-sensitive content and inserting the associated locale-sensitive content back into the compiled document (column 3 lines 50-55).

Hinks does not expressly disclose *removal and substituting a function call in place of associated locale-sensitive content in the compiled document; and the populating comprises executing the function call in the compiled document.*

However, in an analogous environment, Sun teaches *substituting a function call in place of associated locale-sensitive content in the compiled document* (page 99 under the header “xgettext and msgfmt Utilities” as cited in the above rejection of claim 6) *and the populating comprises executing the function call in the compiled document* (page 98 under the header “gettext() and dgettext() Routines” as cited in the above rejection of claim 6).

All further limitations have been addressed in the above rejection of claim 1.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun’s function call substitution and population in Hinks’ localization system. One of ordinary skill would have been motivated to include nothing

specific to one's language and culture in software development and to provide features that facilitate translation of text into other languages.

As per claims 21 and 22, the above rejection of claim 20 is incorporated. All further limitations have been addressed in the above rejection of claims 3 and 4, respectively.

As per claim 23, the above rejection of claim 20 is incorporated. Further, Hinks discloses *storing the locale-sensitive content in a structured text file* (column 3 lines 9-11. Resource files structured text files.).

As per claim 24, the above rejection of claim 20 is incorporated. Further, Hinks discloses *storing the locale sensitive content in a database file* (column 3 lines 13-16).

As per claim 26, the above rejection of claim 26 is incorporated. Hinks further discloses:

storing one or more translated versions of the locale-sensitive content (column 3 lines 23-24); and

at runtime, retrieving the compiled document and populating the compiled document with a translated version of the locale-sensitive content (column 3 lines 50-55).

As per claim 32, Hinks discloses a system (FIG. 3). Hinks further discloses:

at least one computer-servable document stored in a computer-readable medium, the document being written for a particular locale See column 3 lines 7-9:

First, the **sources** are parsed by the **Export/Import** module to a translatable format.

Also column 5 lines 59-61:

Software system 200, which is stored in system memory 102 and on disk memory 107, includes a kernel or operating system (OS) 240 and a windows shell 250.

a compiler to automatically extract ... locale-sensitive content from the document to produce a compiled document containing locale-independent elements See column 3 lines 7-7:

First, the sources are parsed by the **Export/Import** module to a translatable format.

Also column 3 lines 50-51:

Resources for products to be translated are stored in an **external resource file** in a standard format.

Hinks also discloses obtaining the associated locale-sensitive content and inserting the associated locale-sensitive content back into the compiled document (column 3 lines 50-55).

Hinks does not expressly disclose *removal and substituting a function call in place of associated locale-sensitive content in the compiled document*. All further limitations have been addressed in the above rejection of claim 5.

However, in an analogous environment, Sun teaches *substituting a function call in place of associated locale-sensitive content in the compiled document* (page 99 under the header “xgettext and msgfmt Utilities” as cited in the above rejection of claim 6) *the function call being configured such that, when executed, the function call obtains the associated locale-sensitive content from the data structure and inserts the associated*

locale-sensitive content back into the compiled document (page 98 under the header “gettext() and dgettext() Routines” as cited in the above rejection of claim 6).

All further limitations have been addressed in the above rejection of claim 1.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun’s function call substitution and population in Hinks’ localization system. One of ordinary skill would have been motivated to include nothing specific to one’s language and culture in software development and to provide features that facilitate translation of text into other languages.

As per claims 33 and 34, the above rejection of claim 32 is incorporated. All further limitations have been addressed in the above rejections of claims 3 and 4, respectively.

As per claim 35, the above rejection of claim 32 is incorporated. Hinks further discloses *wherein the compiler examines source code in the document to determine, from the source code, whether locale-sensitive content is present* (column 3 lines 7-9).

As per claim 41, Hinks discloses:

A compiler system (FIG. 3) comprising:

a grammar containing rules for structuring source code (column 3 lines 7-9 as cited in the above rejection of claim 1 describes a parser which inherently uses a source code grammar, otherwise it would be unable to match tokens of the source code.);

a call library to store function calls (column 5 lines 26-29 describe use of the Microsoft Windows environment, which inherently contains a call library);
content analyzer to analyze source code in a document written for a particular locale and to utilize the grammar to determine whether the source code contains locale-sensitive content that is specific to the particular locale (column 3 lines 7-9 as cited in the above rejection of claim 1),

Hinks does not expressly disclose *the content analyzer being configured to the locale-sensitive content in the source code.*

However, in an analogous environment, Sun teaches *the content analyzer being configured to <manipulate> the locale-sensitive content in the source code with associated references to the replaced locale-sensitive content* (page 99 under the header “xgettext and msgfmt Utilities”, and page 98 under the header “gettext() and dgettext() Routines” as referenced in the above rejection of claim 6).

All further limitations have been addressed in the above rejection of claim 1.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun’s reference replacement in Hinks’ localization system. One of ordinary skill would have been motivated to include nothing specific to one’s language and culture in software development and to provide features that facilitate translation of text into other languages.

As per claim 42, the above rejection of claim 41 is incorporated. Hinks further discloses *wherein the locale-sensitive content is placed in a separate file*, (column 3 lines 13-16).

Hinks does not expressly disclose *and the references comprise a pointer to that file*.

However, Sun teaches a reference to a file containing locale-sensitive content (page 97 under “Text Databases (Text Domains)”.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun’s reference pointer with Hinks’ content file. One of ordinary skill would have been motivated to retrieve text in the local language from a program with placeholders for a localizer to put each string’s translation.

As per claim 43, the above rejection of claim 41 is incorporated. Hinks further discloses *wherein the locale-sensitive content is placed in a separate data structure* (column 3 lines 13-16 as cited in claim 5).

Hinks does not expressly disclose references that comprise function calls.

However, Sun teaches *the references comprise function calls that, when executed, obtain the associated locale-sensitive content from the data structure and insert the associated locale-sensitive content into the source code* (page 99 under the header “xgettext and msgfmt Utilities” and page 98 under the header “gettext() and dgettext() Routines”).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun's function calls to reference Hinks' data structure. One of ordinary skill would have been motivated to retrieve text in the local language from a program with placeholders for a localizer to put each string's translation.

As per claim 49, Hinks discloses:

A system (FIG. 3) comprising:

compilation means for compiling a computer-servable document written for a particular locale to extract ... any locale-sensitive content (column 3 lines 1-4, and 7-9 as cited in the above rejection of claim 1), the compilation means producing a compiled document with locale-independent elements (FIG. 3 element 377 and column 8 lines 30-34 as cited in the above rejection of claim 13); and

storage means for storing the locale-sensitive content extracted from the computer-servable document in a data structure separate from the compiled document (column 3 lines 13-16 as cited in the above rejection of claim 5).

All further limitations have been addressed in the above rejection of claim 1.

Hinks does not expressly disclose substitution with a reference. However, Sun teaches all further limitations as have been addressed in the above rejection of claim 13. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun's reference substitution in Hinks' localization system. One of ordinary skill would have been motivated to include nothing specific to one's language

and culture in software development and to provide features that facilitate translation of text into other languages.

As per claim 51, the above rejection of claim 49 is incorporated. All further limitations have been addressed in the above rejection of claim 18.

As per claim 52, the above rejection of claim 49 is incorporated. All further limitations have been addressed in the above rejection of claim 19.

As per claim 55, Hinks discloses:

One or more computer-readable media (column 5 line 44: “main memory”)
comprising computer-executable instructions that, when executed, direct a computer to:
examine source code in a document written for a particular locale (column 3 lines 7-9: “First, the sources are **parsed** by the Export/Import module to a translatable format.”);
extract ... any locale-sensitive content from the source code (column 3 lines 1-4: “The Export/Import module itself includes a parsing engine to **extract** strings and translatable information from application programs.”);

store the locale-sensitive content in a separate file (column 3 lines 13-16: “From there, Export/Import (EXPIMP) module parses the resource

file into a Translation Table, which is typically stored as a database table.”).

Hinks does not expressly disclose: *substitute, in a same operation as the extraction and removal, in place of the locale-sensitive content in the document, function calls which when executed at runtime, re-supply the locale-sensitive content to the document.*

However, Sun teaches: *substitute, in a same operation as the extraction..., in place of the removed locale-sensitive content in the document, function calls which when executed at runtime, re-supply locale-sensitive content to the document*(page 99 under the header “xgettext and msgfmt Utilities” and page 98 under the header “gettext() and dgettext() Routines). All further limitations have been addressed in the above rejection of claim 1.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Sun’s reference substitution in Hinks’ localization system. One of ordinary skill would have been motivated to include nothing specific to one’s language and culture in software development and to provide features that facilitate translation of text into other languages.

10. Claims 7, 8, and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sun in view of in view of Lyapustina.

As per claim 7, Sun discloses:

A method comprising:

examining source code in a document written for a particular locale See page 97

“Text Databases”:

To facilitate internationalization of text, a process exists (see `gettext()`, later in this section) to collect all text visible to the user into a file called a portable object file. The portable object file contains the native language strings from a program and placeholders for a localizer to put each string’s translation.

determining, from the source code, locale-sensitive content that is specific to the particular locale (page 97 as cited above, “native language strings”);

extracting ... the locale-sensitive content from the source code See page 97 as cited above, “collect all text”; also page 100:

If you then use the GXV code generator to generate the C source code, the interface’s text strings get linked with the `gettext()` or `dgettext()` function call. These calls are used to look up the strings in other languages, or locales.

Locale-sensitive content is manipulated through the process of linking the function calls in the code. The function calls use the text strings as keys to a text database of translations for other locales. As they are used as keys to a database and are no longer directly available to the original source code, the strings are no longer locale-sensitive.);

storing the locale-sensitive content in a separate file (page 97 as cited above, “portable object file”); and

inserting, in a same operation as the <manipulation>, into the document in place of the removed locale-sensitive content, a reference to the locale-sensitive content in the separate file See page 98 under the header gettext() and dgettext() Routines:

Two similar routines are available to a developer for retrieving translated text. One, `gettext()`, assumes a text domain has already been specified...

Also page 99 under the header xgettext and msgfmt Utilities:

Once Devguide or a developer has inserted gettext() function calls around all user visible text in an application, xgettext can be run on the source files to produce the portable object files.

wherein the reference comprises a function call that, when executed, obtains the locale-sensitive content from the separate file (page 98 “retrieving translated text”).

All further limitations have been addressed in the above rejection of claim 1.

As per claim 8, the above rejection of claim 7 is incorporated. Sun further discloses *wherein the locale-sensitive content comprises natural language text* (page 97 “native language strings”).

As per claim 10, the above rejection of claim 7 is incorporated. Sun further discloses *wherein the storing comprises storing the locale sensitive content in a structured text file* (page 99 “portable object file”).

11. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Sun and Lyapustina as applied to claim 7 above, and further in view of U.S. Patent Application Publication US 2002/0107684 to Gao, filed Feb. 7, 2001 (hereinafter referred to as “Gao”).

As per claim 9, Sun does not expressly disclose ascertaining and identifying type. However, in an analogous environment, Gao teaches internationalizing software using a semantic analysis phase:

ascertaining a type of code elements using a grammar for the source code (page 3, paragraph 51); and

identifying, based on the type of the code elements, any locale-sensitive content delimited by the code elements (page 3, paragraph 52).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Gao's type analysis and identification determination in Sun's internationalization method. One of ordinary skill would have been motivated to detect potential internationalization problems using language grammar rules.

12. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over Sun and Lyapustina as applied to claim 7 above, and further in view of Hinks.

As per claim 11, Sun does not expressly disclose storing content in a database file. However, Hinks teaches the use of a database file for storage of locale sensitive content (column 3 lines 13-16).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to store Sun's localization data in Hinks database file. One of ordinary skill would have been motivated to store localization data in an easily searchable data structure.

13. Claims 39 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hinks, Sun, and Lyapustina as applied to claim 32 above, and further in view of Gao.

As per claim 39, the above rejection of claim 32 is incorporated. Further, Hinks does not expressly disclose a runtime manager.

However, Gao teaches the use of an “Integrated Translation Environment” which populates web documents with locale-sensitive content prior to serving (page 5 paragraph 0135).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Gao’s web localizer with Hinks’ internationalization method. One of ordinary skill would have been motivated to present localized content in a document.

As per claim 40, the above rejection of claim 32 is incorporated. Hinks further discloses a translated version of the locale-sensitive content (column 3 lines 23-24).

Hinks does not expressly disclose a runtime manager.

However, Gao teaches the use of an “Integrated Translation Environment” which populates web documents with locale-sensitive content during runtime (page 5 paragraph 0135).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Gao’s web localizer with Hinks’ internationalization method. One of ordinary skill would have been motivated to present localized content in a document when it is presented.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to J. Derek Rutten whose telephone number is (571) 272-3703. The examiner can normally be reached on T-F 6:00 - 4:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

jdr



TUAN DAM
SUPERVISORY PATENT EXAMINER